Resumen

Se ha detectado que muchas empresas manufactureras utilizan un software para la gestión y seguimiento de la producción. Este software está disponible para los trabajadores en el lugar de trabajo, pero estos datos no son accesibles desde su casa o cuando realizan viajes al extranjero.

A lo largo del proyecto se ha desarrollado una **AppMovil** que permite tener acceso a la gestión de la fabricación desde el teléfono móvil.

Se ha programado un prototipo de AppMovil que permite al usuario **controlar la fabricación** de los diferentes productos de una empresa, y poder **visualizar la ejecución** de esta fabricación.

Para el correcto funcionamiento de la AppMovil, también se ha desarrollado una AppServer, en contacto directo con el software que actualmente gestiona la producción.

La programación de la AppMovil se ha realizado en lenguaje python, con un framework llamado Kivy. La programación de la AppServer se ha realizado en lenguaje PHP. La comunicación entre ambos se realiza a través del protocolo web HTTP.

Sumario

1. Introducción

En los últimos tiempos, la necesidad de estar informados y que esta información sea cada vez accesible de manera inmediata, es una constante en los avances tecnológicos de las últimas décadas. Sólo hay que observar la evolución de la tradicional prensa escrita, que se convirtió en la nueva prensa digital, y que en la actualidad trata de transmitir las noticias de manera sencilla y rápida, por ejemplo, vía Twitter.

Esta necesidad de **información inmediata** no está implementada de forma extensa en la comunidad industrial.

Las empresas industriales de manufactura tienen una importante carga de trabajo relacionada con el proceso industrial que lleva a cabo la fabricación de sus productos. La fabricación de estos productos genera una serie de información: tiempos, materiales, mermas, rechazos, etc.

Existe un gran número de empresas industriales que recogen estos datos relativos a la fabricación de sus productos. Se pueden recoger de varias maneras, las más comunes que se encuentran son: apuntando manualmente sobre papel o sobre soporte informático, recogiendo los datos de manera automática, o una mezcla de ambas.

En el caso que la recogida de datos se realice sobre un sistema informático, éste recibe el nombre inglés de **Manufacturing Execution System (MES)**.

Existe una gran cantidad de empresas que utilizan sistemas MES. Un factor común en todas ellas es que este sistema está disponible únicamente a través de un terminal PC, situado en la propia planta industrial.

Las personas encargadas del control y/o gestión de la producción no tienen acceso durante todo el día al ordenador que les permite dicho control.

Llegado a este punto, se ve la necesidad de dar un paso más en la **accesibilidad de esta información**, y mostrar toda esta información a través del teléfono móvil, el cual es accesible desde muchos otros lugares que no sea la planta productiva.

1.1. Qué es el MES

Como se ha comentado anteriormente, **MES** proviene de las iniciales de las palabras Manufacturing Execution System, sistema de ejecución de la fabricación. Este sistema nos permite hacer **seguimiento de la fabricación** en una planta productiva en la que se lleven a cabo procesos industriales.

A la vez que permite el seguimiento de la producción actual, el sistema también **almacena los datos adquiridos** para poder ser procesados con posterioridad. Este potencial es muy importante para la eficiencia de la empresa a largo plazo. Permite saber que procesos son los más eficientes, y así determinar las fortalezas de la fábrica. De la misma manera que permite detectar las operaciones ineficientes para actuar sobre ellas.

El MES es un software que funciona en paralelo con el ERP, sistema de gestión o de planificación de la empresa. Aunque no es imprescindible la existencia de un ERP para su funcionamiento, en la práctica, toda empresa que implementa un MES en producción también utiliza un sistema de gestión.

La relación del MES con el ERP aparece en el momento que el ERP genera unas **necesidades productivas** y éstas se planifican en las diferentes máquinas o estaciones de trabajo. La información que proporciona la planificación es la necesaria para hacer funcionar al MES. Es decir, el MES necesita saber qué fabricar y con qué recursos (máquinas, materiales y mano de obra).

El MES devuelve al ERP la información relativa a los **costes, materiales y tiempos**, para poder determinar el coste del producto y hacer las regularizaciones pertinentes en los stocks de los materiales y productos fabricados.

En la Fig. 1 se observa el esquema de comunicación anteriormente descrito.

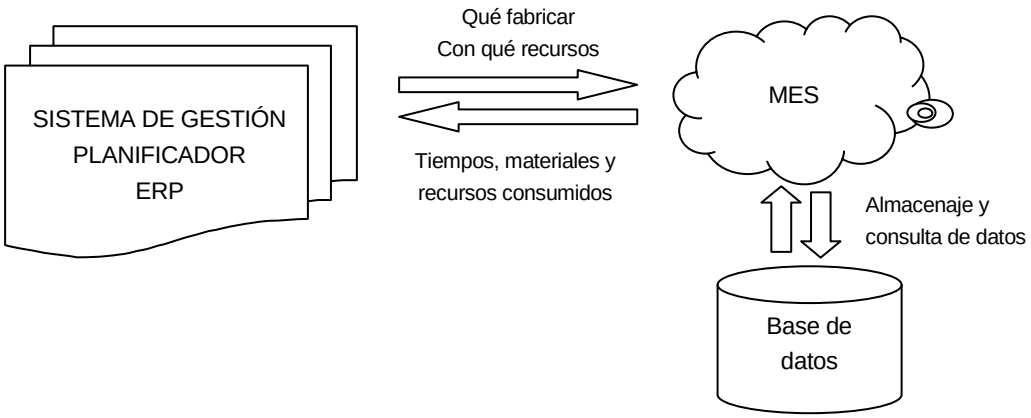


Fig. 1. Relación del MES con el ERP

1.2. Objetivos del proyecto

El objetivo del proyecto es diseñar el prototipo de una **estructura de comunicación que permita la utilización del MES a través de un dispositivo móvil**, sin necesidad de estar presente en la planta de producción.

A continuación, se exponen otros objetivos secundarios a cumplir por el proyecto.

- Acceso fácil, cómodo y rápido a las funcionalidades, para fomentar su utilización y evitar la procastinación.
- Implementación de nuevas funcionalidades de manera rápida y económica.
- Flexibilidad para adquirir los datos cualquiera sea el modelo de MES implantado en el proceso productivo.
- Diseño en función de la normativa ANSI/ISA-95 de International Society of Automation para el desarrollo de la interface entre la empresa y los sistemas de control.

1.3. Requerimientos iniciales

Es necesario que la planta productiva en la que se vaya a implantar el proyecto disponga de un **MES previamente implantado**, que recoja los datos proporcionados por el proceso industrial.

Dentro de las características del MES, se pueden observar variantes. Aquellos que recogen los datos productivos proporcionados por las máquinas, los que recogen los datos por las personas, o una combinación de ambas. Uno de los objetivos del proyecto es la flexibilidad para poder trabajar con cualquiera de estos sistemas implementados.

1.4. Alcance del proyecto

Los sistemas MES que utiliza la industria tienen multitud de variantes y configuraciones, intentar crear un único software capaz de abarcar a todas estas es una tarea realmente compleja. Es común encontrarse con sistemas MES que monitorizan múltiples líneas productivas independientes con una estación de trabajo cada una, o de otro modo, una sola línea productiva con múltiples estaciones de trabajo.

Por otro lado, los datos captados por el MES estarán almacenados en alguno de los tipos de bases de datos que existen en el mercado.

Al tener en cuenta estas variantes, el proyecto se limitará a replicar una base de datos SQL básica que contenga la información que cualquier MES comercial almacena.

De este modo, no se entrará en detalles que pueden ser interesantes para algunas empresas en particular, como podría ser el seguimiento del plan de mantenimiento de una máquina, o la monitorización del ritmo de una prensa. Este tipo de características específicas se determina que pueden ser añadidas en próximos desarrollos.

Al no realizar el desarrollo para una planta concreta, el plan de implantación y formación tampoco se llevará a cabo.

El **prototipo de App** a diseñar será una primera versión, sobre la cual sea posible el desarrollo de nuevas funcionalidades.

Las funcionalidades genéricas a desarrollar son la **visualización y modificación de la planificación** y la **visualización de la producción**.

2. Visualización general de la AppMovil

La visualización de la App será el punto de acceso al MES cuando no se esté presente en la planta industrial, o estando presente, no se tenga acceso a un terminal para conectarse al mismo. Con esta premisa, hay que tener en cuenta que la App será de **uso intensivo**.

Con el fin de facilitar e incentivar su uso, es necesario que se diseñe teniendo en cuenta las premisas de **funcionalidad y usabilidad**. Debe ser organizada e intuitiva de utilizar.

Se estructurará según el estándar ISA-95, en la Fig. 2 se puede observar las funcionalidades recogidas en la norma.

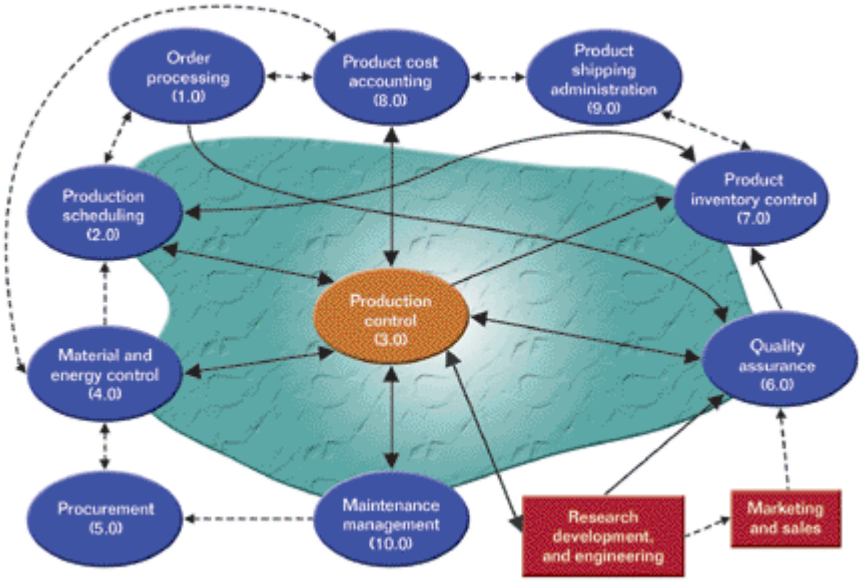


Fig. 2. Representación gráfica de la funcionalidad según norma ISA-95.

A continuación, se observa la estructura de las diferentes funcionalidades que se desarrollarán en el proyecto.

1. Planificación
   1.1. Órdenes de fabricación (visualizar y modificar)
2. Producción
   2.1. Seguimiento de las operaciones realizadas.

Se puede observar que la cantidad de funcionalidades que se desarrollarán no es la misma que las funcionalidades incluidas en la norma ISA-95 como se ha comentado anteriormente, debido a la falta de relación coste/utilidad para muchas de las empresas.

Se seguirán **criterios de usabilidad** para la interfaz de usuario (UI, user interface). Entre los que se encuentran la *facilidad de aprendizaje*, la *eficiencia* y la *perdurabilidad en la memoria*. En resumen, se necesita que el usuario sea capaz de aprender por sí mismo el funcionamiento de la App (***facilidad***), sin necesidad de complicados manuales. Una vez el usuario sabe utilizarla, es necesario que se manejen con soltura y las tareas sean rápidas de realizar (***eficiencia***). Por último, que tras un tiempo de inactividad con la App, el usuario sea capaz de retomar su uso con facilidad (***perdurabilidad***). [1]

Fig. 3. Pantalla inicial de la AppMovil

3. Estructura de funcionamiento

La AppMovil que se diseñe, debe mostrar datos almacenados en el servidor de datos de conectado a la planta industrial. Así, esta AppMovil debe estar conectada con el servidor de datos.

En el servidor situado en la planta industrial, conectado a los dispositivos de recogida de **datos del proceso industrial**, dispone de los datos proporcionados por estos dispositivos. Para hacerlos **accesibles a la AppMovil** deben ser proporcionados a ésta. Por este motivo, se necesita implementar una AppServer que recoja los datos del propio servidor y estén disponibles para la AppMovil.

Una vez se dispone de la AppMovil y el AppServer, éstos deben comunicarse entre sí para compartir los datos. Esta comunicación se realiza a través de un **canal de comunicación** que se rige por unas normas de funcionamiento y una determinada estructura, llamado protocolo de comunicación.

Antes de entrar en detalle en el funcionamiento de cada uno de los módulos, se determinará el flujo de información entre ellos.

3.1. Flujo de información

En cada una de las **interacciones** que realiza el **usuario con la AppMovil** se establece un flujo de información entre los diferentes procesos que componen el proyecto.

Aunque el objeto del proyecto lo constituye la **AppMovil** y la **AppServer**, entre los procesos necesarios para funcionar también se encuentra el **servidor web**, el motor de la **base de datos** y el **MES**.

El MES se considera como un único proceso dentro del proyecto, aunque en la práctica está compuesto por varios de ellos.

En la Fig. 4 se observa que la **AppMovil se comunica con el servidor web**, esta comunicación se realiza mediante protocolo HTTP, que más adelante se explicará.

El servidor web, a su vez, se comunica con la AppServer i el motor de la base de datos. Este pack de procesos funciona sobre un único hardware, con un sistema operativo Linux (Lubuntu). Como servidor web se ha optado por Apache, como base de datos: MySQL, y la AppServer en HTML+PHP. Este conjunto de software se le denomina LAMP, nombre que proviene de las iniciales de: Linux, Apache, Mysql y Php.

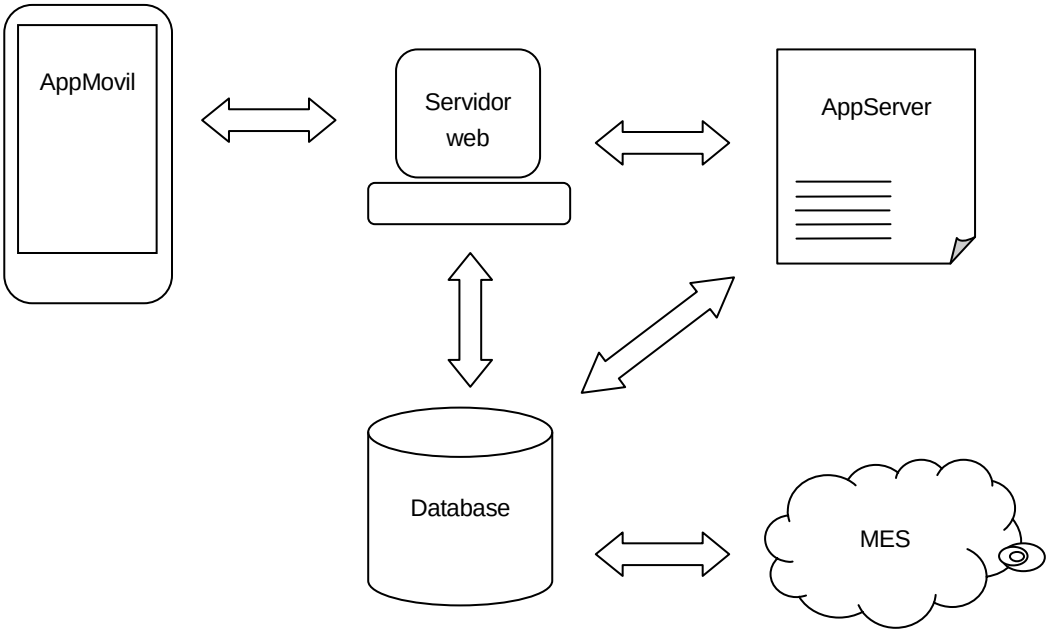


Fig. 4. Flujo de información entre los procesos implicados

La AppMovil se conecta con el Servidor web a través de **peticiones HTTP**. El servidor enviará información a la AppMovil en datos estructurados en **formato JSON**.

3.2. Estructura de la información

Los datos con los trabajará la AppMovil se sincronizarán con los datos del MES. Para ello, la estructura de los datos debe ser compatible, tanto en formato, como en estructura.

Así, se estudiará la **estructura de los datos** con los que trabaja un MES genérico, que a su vez, recoge los datos del ERP. Esta información se almacena en una base de datos.

Antes de analizar la estructura de la base de datos, se recuerda como se expresa el proceso productivo de un producto de forma gráfica.

La fabricación de un producto se expresa con un **escandallo** de su **proceso productivo**, con las **operaciones** que lo componen y los **materiales** necesarios en cada operación. En la Fig. 5 se observa un ejemplo de escandallo de la fabricación de una mesa.

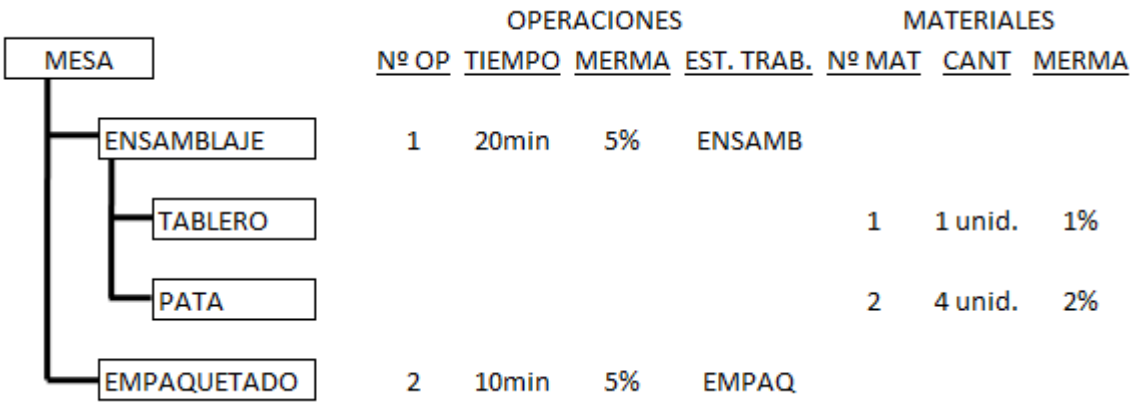


Fig. 5. Ejemplo de escandallo de una mesa, con operaciones y materiales.

La base de datos es estructurada, y los datos están almacenados en tablas. La información reflejada en la Fig. 5 se almacena en la base de datos con dos tablas, la primera contiene datos referente a las operaciones, y la segunda con datos de los materiales.

Siguiendo el ejemplo de la Fig. 5, las tablas de datos son las que se muestran en la Fig. 6 (lista de operaciones) y Fig. 7 (lista de materiales).

list_operac

Id	ref_padre	operacion	descr	est_trab	tiempo_teor	merma_tiempo
1	MESA	1	ENSAMBLAJE	ENSAMB	20	5
2	MESA	2	EMPAQUETADO	EMPAQ	10	5
(Nuevo)						

Fig. 6. Ejemplo de lista de operaciones

list_mat

Id	ref_padre	operacion	material	ref_hijo	cant_hijo_teor	merma_hijo
1	MESA	1	1	TABLERO	1	1
2	MESA	1	2	PATA	4	2
(Nuevo)						

Fig. 7. Ejemplo de lista de materiales

Estas dos tablas incluyen toda la información relativa al proceso productivo de cualquier producto que se fabrique en la empresa.

Para almacenar la información de la producción se necesitan otras tablas, que se detallan a continuación:

Artículos. Incluye información referente a las referencias con las que trabaja la empresa, tanto productos de venta, como productos de compra o semi-elaborados. Algunos campos

son: referencia, descripción, lote de venta, lote de fabricación, coste de fabricación teórico, coste de la última fabricación, familia de producto, plazo de entrega, etc.

Estaciones de trabajo (ET). Es el lugar donde se lleva a cabo una operación, por ejemplo, una máquina o una mesa de trabajo. En esta tabla se almacena los datos relativos a estas estaciones de trabajo.

Órdenes de fabricación (OF). Cada vez que se desea producir una referencia, se crea una OF. Al crear una OF, se determina el producto a fabricar, la cantidad y la fecha teórica de fabricación, estos datos se insertan en esta tabla.

Operaciones de la OF. Esta tabla contiene la información extraída del escandallo del producto a fabricar, de la tabla Lista de operaciones (Fig. 6)

Materiales de la OF. Contiene los materiales que se van a utilizar en la fabricación de una OF específica. Los datos los extrae de la tabla Lista de materiales (Fig. 7)

En el mismo momento que se crea una OF, se insertan los datos en la tabla de Órdenes de Fabricación, Operaciones de la OF y Materiales de la OF.

La Fig. 8 representa las relaciones que existen entre cada una de las tablas anteriormente descritas.

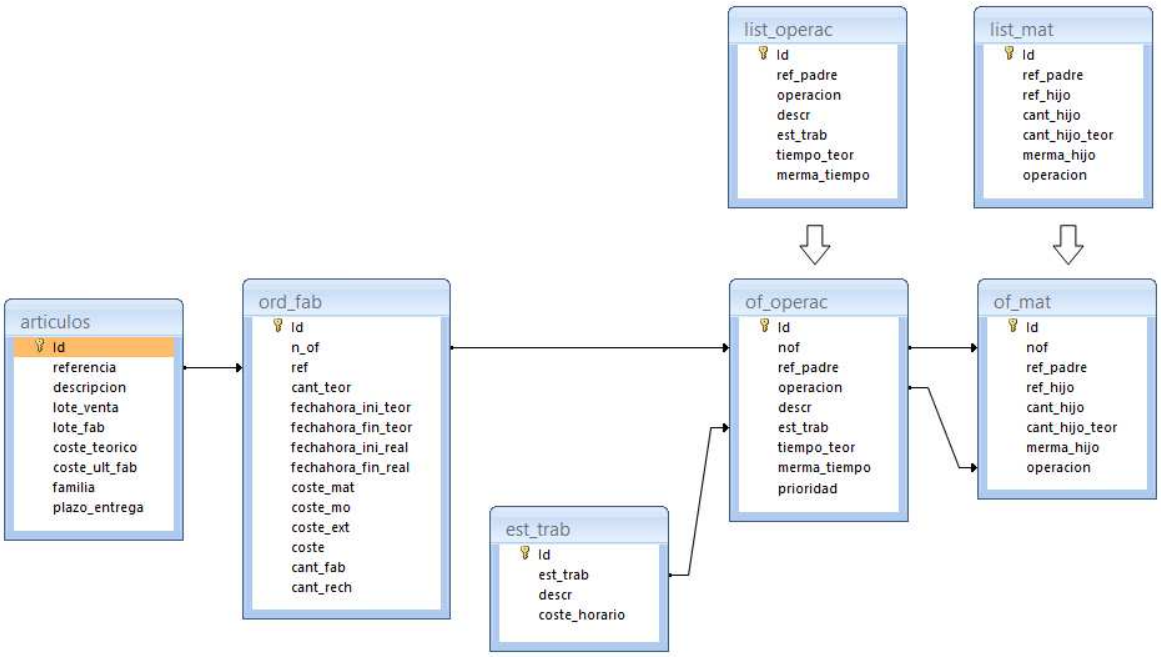


Fig. 8. Relaciones entre las diferentes tablas de la base de datos del MES

4. AppMovil

La AppMovil es el software que necesita de un dispositivo programable para funcionar. En el mercado, existen multitud de dispositivos que pueden ser programados con este propósito. De la misma manera, cada uno de estos dispositivos puede ser programado en los diferentes lenguajes de programación que soporte.

4.1. Dispositivo

Cómo se ha definido en los objetivos, se requiere que el dispositivo esté en comunicación con los datos de la planta industrial. También se necesita que su uso sea fácil y cómodo.

Los dispositivos que se han analizado para desarrollar la aplicación son los smartphones, tablets, portátiles, ordenador sobremesa y open hardware (cómo rasperry pi y arduino).

En la Tabla 1 se observa una comparativa entre éstos, evaluando diferentes parámetros y valorándolos sobre una escala del 1 al 5, según se estima.

Característica (valoración)	Smartphones	Tablet	Portátil Netbook	PC sobremesa	Open hardware	Disp. industrial
Coste (5)	3	2	1	1	5	1
Movilidad (5)	5	4	3	1	3	5
Uso extendido en el público general (5)	5	5	5	5	1	1
Visualización (3)	2	4	5	5	4	3
Evaluación final	75	75	70	60	65	50

Tabla 1. Evaluación de varios dispositivos analizados.

Teniendo en cuenta la evaluación anterior, se estima que el dispositivo adecuado para desarrollar la aplicación móvil es el smartphone y la tablet.

Según los últimos estudios de mercado, entre mayo y agosto de 2015, el sistema operativo más utilizado en los smartphones es Android, con una cuota de mercado de 89,4%. Seguido de iOS, con un 6,6%. [2]

Por lo tanto, para el desarrollo del prototipo se tomará un **smartphone Android**, en concreto, el modelo Motorola MotoG con la versión 5.1 de Android.

4.2. Programación

Android es un sistema operativo pensado para ser programado en lenguaje Java con Android Software Development Kit (Android SDK). Aún así, puede ser que por algún motivo convenga programar en algún otro lenguaje, como C/C++ con el Native Development Kit (NDK). [3]

En nuestro caso, uno de los requisitos principales del proyecto es la facilidad de desarrollo de nuevas funcionalidades. Para esto, se utilizará el **framework** llamado **Kivy**, que hace que la programación de interfaces de usuario sea cómoda y rápida para el programador.

Kivy es un framework de código abierto, desarrollado en **Python** para el desarrollo de aplicaciones móviles multitáctiles con interfaz natural de usuario (NUI).

La escalabilidad del proyecto es importante, se necesita poder implementar en un futuro nuevos módulos de funcionalidades. Con el fin de poder satisfacer esta necesidad, se necesita una programación limpia i ordenada. De esta manera, se ha utilizado el sistema de programación de **Modelo-Vista-Controlador** (MVC). [4]

El MVC estructura el software en 3 unidades de código diferenciadas: modelo, vista y controlador. En la Fig. 9 se observa la estructura de comunicación entre cada uno de ellos.

Modelo. Gestiona los accesos a la información. Envía a la vista la información que le es solicitada. Constituye la lógica de negocio, es decir, las especificaciones de la aplicación.

Vista. Representa la información facilitada por el modelo en un formato agradable para interactuar.

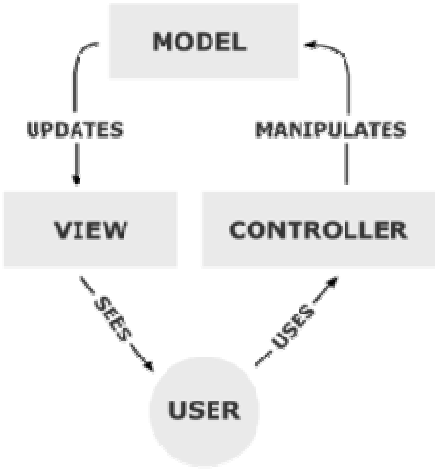


Fig. 9. Estructura MVC

Controlador. Envía al modelo las peticiones del usuario. También invoca a la vista a representar dichas peticiones, si es necesario.

El prototipo de AppMovil, se compone de 3 módulos: la **pantalla inicial**, el **planificador** y la **visualización de la producción**. En cada uno de estos módulos se programa según la estructura MVC.

4.3. Funcionamiento

4.3.1. Pantalla inicial

Al inicial el software, el controlador comunica al modelo que se debe iniciar la aplicación, éste, indica a la vista que represente los botones que enlacen con las diferentes funcionalidades (planificación, producción y salida).

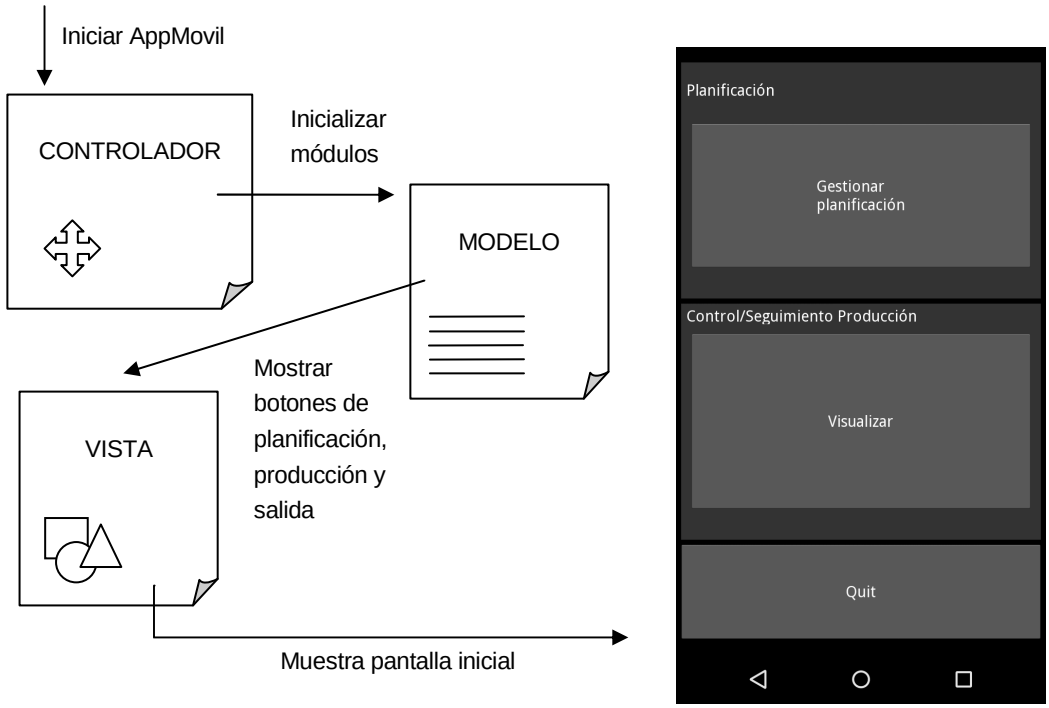


Fig. 10. Comunicación entre Modelo-Vista-Controlador al iniciar la aplicación.

Al pulsar sobre el botón de planificación, el controlador capta el evento. A continuación, manda una orden al modelo para cargar el módulo de planificación. Luego, el modelo manda la orden de visualizar la pantalla de planificación a la vista. (Fig. 11)

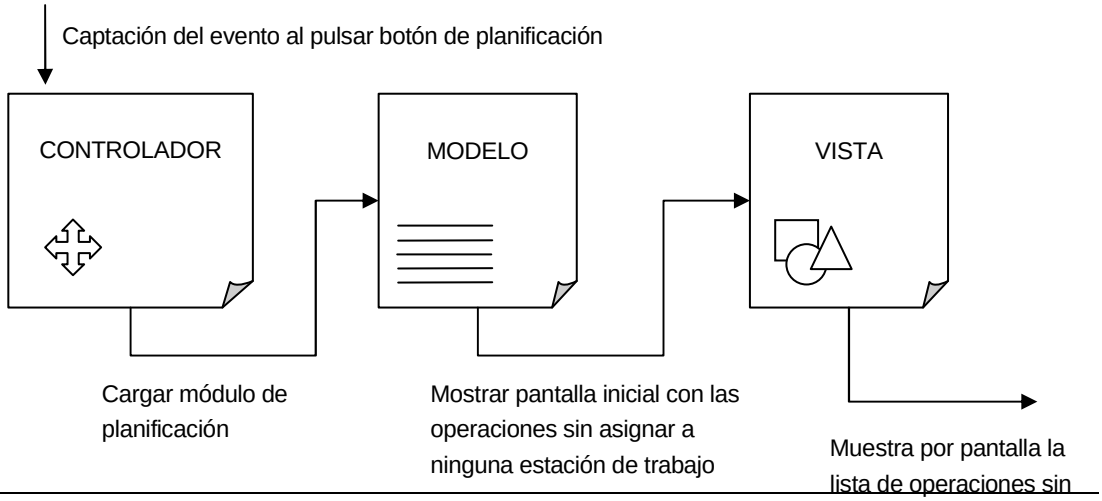


Fig. 11. Comunicación entre Modelo-Vista-Controlador al pulsar el botón de planificación

4.3.2. Módulo Planificación

El módulo de planificación dará acceso a la **visualización de las operaciones** de las órdenes de fabricación creadas por el MES/ERP.

Esta visualización se hará a través de las diferentes **estaciones de trabajo** que se hayan configurado en el MES. Si una operación no está asignada a ninguna estación de trabajo, éstas se podrán ver a través de la estación de trabajo “Sin Asignar”.

Este módulo también permitirá hacer modificaciones en las operaciones, se podrá **asignar una estación de trabajo a una operación** que no tenga ninguna asociada, o se podrá cambiar la estación de trabajo asignada por otra.

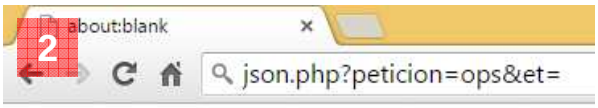
Por otro lado, también se podrá **cambiar el orden o prioridad** de cada una de las operaciones, de esta manera se afectará directamente en el trabajo a realizar en la fábrica.

Las acciones a realizar en la planificación tienen una **comunicación intensiva con el servidor web** (AppServer y base de datos). Cada vez que se cambie la estación de trabajo, o el orden de una operación, la AppMovil se comunicará con el servidor web, y éste, a su vez con la base de datos del MES para su actualización.

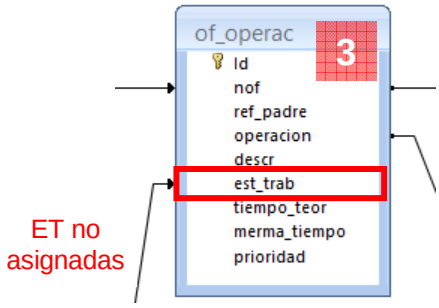
En la Fig. 12 se muestra las visualizaciones y los procesos internos al abrir el módulo de planificación:

1. Se presiona sobre "Gestionar planificación".

2. La AppMovil pide al servidor las operaciones con la ET sin asignar.



3. La AppServer recibe la petición y hace la consulta en la base de datos.



4. La AppServer envía los datos a la AppMovil

5. Se abre la pantalla de planificación, mostrándose las operaciones "Sin Asignar". La operación "bbbbbaa" no está asignada a ninguna estación de trabajo.

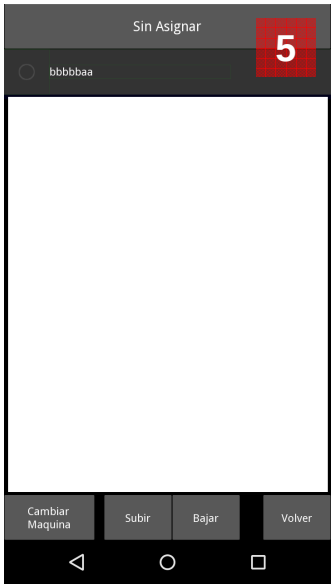
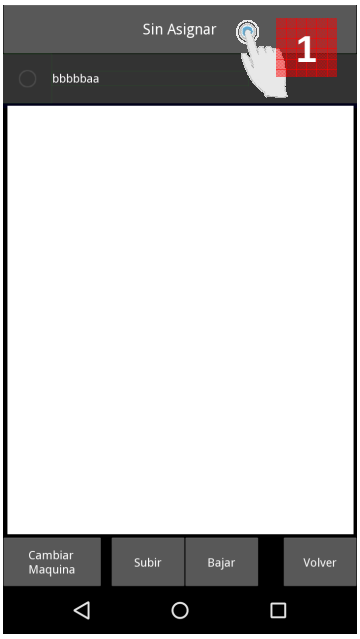


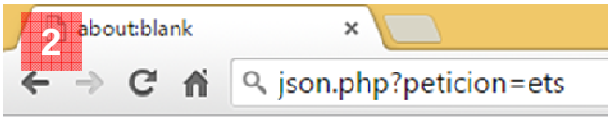
Fig. 12. Funcionamiento del módulo de planificación al iniciarse.

En la Fig. 13 se muestran las comunicaciones internas entre AppMovil, servidor, AppServer y base de datos al desplegar el menú de máquinas.

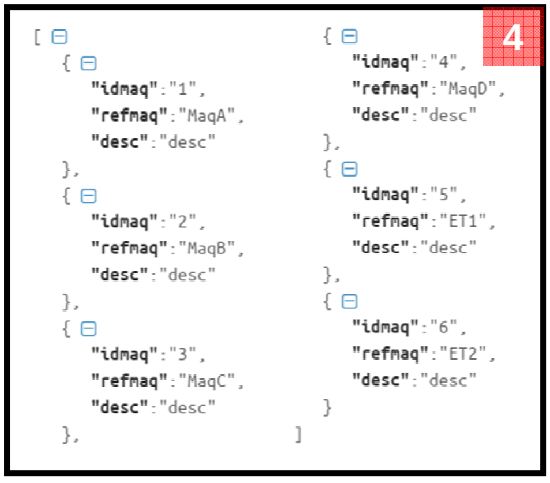
1. Se pulsa sobre el desplegable de máquinas:



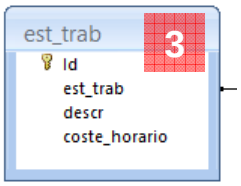
2. La AppMovil pide al servidor un listado de las estaciones de trabajo.



3. La AppServer envía los datos a la AppMovil.



4. La AppServer recibe la petición y hace la consulta de las estaciones de trabajo sobre la base de datos.



5. Se muestra el desplegable de máquinas.

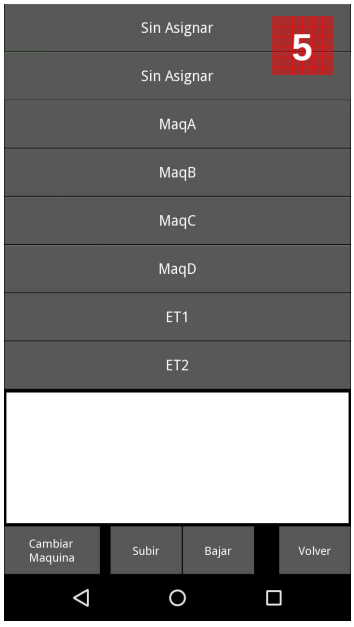


Fig. 13. Funcionamiento del desplegable de máquinas.

4.3.2.1. Funcionalidad subir/bajar prioridad

Los botones de *subir* y *bajar* modifican la prioridad de la operación en la estación de trabajo.

La prioridad de las operaciones determina el orden de fabricación que mostrará el MES al operario en la planta productiva. Las operaciones con prioridad más baja serán las primeras que se fabriquen.

De esta manera, cuando se pulsa uno de los botones de de subir/bajar, se quiere modificar el campo prioridad de la tabla *of_operac* (ver Fig. 8, pág. 14) que tiene asignado la operación

Es importante **garantizar que cada operación tiene asignada una prioridad**, y que la prioridad no está repetida en la estación de trabajo a la que está asignada. En caso que existieran dos operaciones con la misma prioridad, no se podría saber cuál de ellas se fabricaría antes.

Así, al modificar la prioridad de la operación que se desea "*subir*" o "*bajar*", se debe garantizar que el resto de operaciones de la estación de trabajo también modifican su prioridad en caso que sea necesario.

Por ejemplo, en el caso de la Fig. 14. En el estado inicial, las operaciones tienen asignadas las siguientes prioridades:

- Hacer AAA → prioridad 1
- Hacer BBB → prioridad 2
- aaaaaaa → prioridad 3
- bbbbaaa → prioridad 4

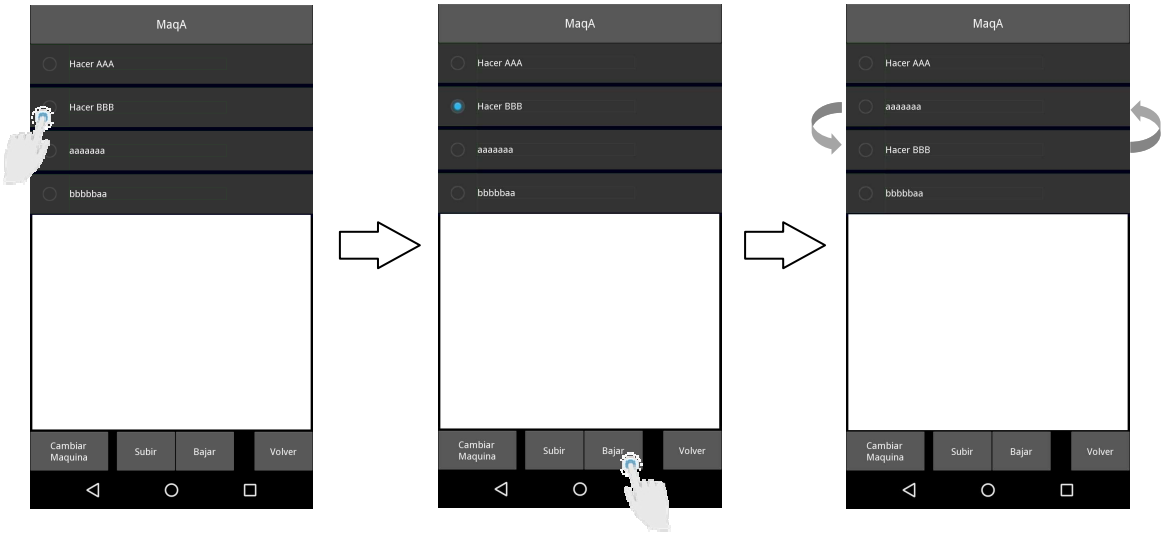


Fig. 14. Ejemplo de funcionalidad subir/bajar prioridad.

Al seleccionar la Operación “Hacer BBB” y a continuación pulsar el botón “Bajar”, implica que la operación seleccionada pase de prioridad 2 a prioridad 3. Además, para garantizar la exclusividad de prioridades dentro de una estación de trabajo, la operación que anteriormente tenía asignada la prioridad 3 pase a tener prioridad 2. En la Fig. 14 se observa la esta funcionalidad.

Un último detalle respecto la funcionalidad subir/bajar. Cuando se selecciona la operación con prioridad 1 y se intenta “subir”, la App no realizará ningún cambio de prioridades, pues si se asignara la prioridad 0, las prioridades no serían correlativas. Lo mismo ocurre al intentar “bajar” la operación con el valor de prioridad más elevado, en el caso del ejemplo anterior, 4.

4.3.2.2. Funcionalidad Cambiar Máquina

El botón "*Cambiar Máquina*" permite asignar una nueva estación de trabajo a una operación.

En la base de datos, este cambio se produce modificando el campo *est_trab* de la tabla *of_operac* (ver Fig. 8, pág. 14).

De la misma manera que en la funcionalidad subir/bajar, es necesario **mantener la integridad de las prioridades** al cambiar las operaciones de máquina. Es decir, si una operación tiene asignada la prioridad X en la estación de trabajo A, al asignarle otra estación de trabajo, la operación debe cambiar la prioridad X por la que le corresponda en la nueva ET.

Por defecto, al cambiar una operación a una nueva ET, se le asignará un entero superior al máximo valor de prioridad que existe en la nueva ET.

$$Prioridad_{operación} = Max(Prioridad_{nuevaET}) + 1$$

Ecuación 1. Prioridad asignada a la operación que modifica la estación de trabajo.

Una vez la operación está en la última posición de la nueva ET, utilizando los botones subir/bajar, se colocará en la prioridad deseada.

Al realizar el cambio de ET, la operación X deja un vacío en las prioridades de la ET de origen. Por ejemplo, si existían 4 operaciones, con las prioridades eran: 1, 2, 3 y 4; al cambiar de ET de una de ellas, podría quedarse con las prioridades: 1, 2 y 4. Y esto ocasionaría problemas al utilizar la funcionalidad de subir/bajar. Para solucionarlo, se recalcula las prioridades de la ET de origen, así, las prioridades serían: 1, 2 y 3.

En la Fig. 15 se observa cómo utilizar esta funcionalidad.

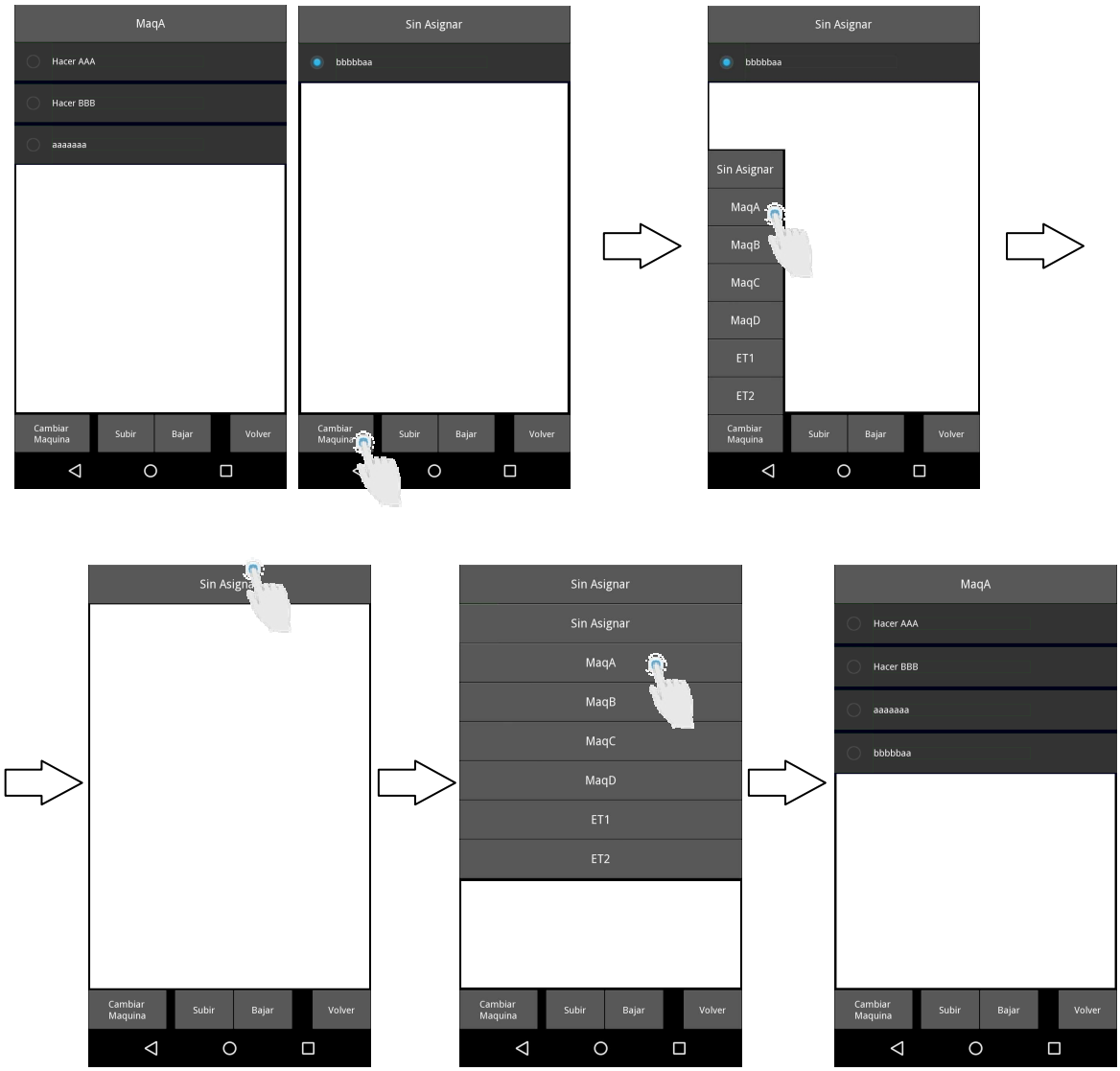


Fig. 15. Ejemplo de la funcionalidad cambiar máquina

4.3.3. Módulo Control/Seguimiento Producción

El módulo de control/seguimiento de la producción permite visualizar el **estado de la fabricación en tiempo real**. Se visualiza el progreso de fabricación de cada orden de fabricación en las diferentes estaciones de trabajo de **forma gráfica**.

Las operaciones que forman una orden de fabricación son secuenciales, es decir, se necesita cumplir el orden de fabricación estricto que está establecido. Según el ejemplo de la Fig. 5, pág. 13 (Ejemplo de escandallo de una mesa, con operaciones y materiales.), es

inviable realizar el empaquetado de la mesa si no se ha realizado anteriormente el ensamblaje.

Cada intervención que se realiza sobre una orden de fabricación recibe el nombre "**bono de trabajo**". Como se ha comentado anteriormente, las órdenes de fabricación están compuestas de operaciones. Así, los bonos de trabajo recaen directamente sobre las operaciones.

En la Fig. 16 se observa un ejemplo de visualización de bonos correspondiente al proceso productivo de la mesa de la Fig. 5.

Existe una OF de 20 mesas. Las operaciones de ensamblaje y empaquetado están asignadas a las ET *ENSAM* Y *EMPAQ*, respectivamente. Se observa que a las 10:00 se inicia un bono correspondiente a la operación *Ensamblar* en el que se han ensamblado 7 mesas, el bono se cierra a las 12:30. Sobre la misma ET se inicia un bono a las 14:00, que se cierra a las 17:15, en el que se han ensamblado 13. En este momento se han ensamblado un total de 20 mesas, las correspondientes a la cantidad de la OF.

En la ET de empaquetado, se inicia el primer bono una vez está iniciado el bono de ensamblaje, cumpliendo la secuencialidad del escandallo. La suma de las cantidades fabricadas en la operación de empaquetado es la misma que la cantidad de la OF.

Fig. 16. Ejemplo de visualización de la producción de una mesa.

5. AppServer

5.1. Necesidad de su uso

El MES es un sistema que **almacena una gran cantidad de información**, pensado para funcionar en diferentes terminales (dispositivos). Es un sistema diseñado para ser centralizado, es decir, un servidor donde se almacena la información y la reparte al resto de terminales.

Los **smartphone son dispositivos con una capacidad de proceso limitada**, así como también lo es su capacidad de almacenamiento. Sería complicado, e ineficiente, almacenar en el smartphone toda la información de las bases de datos del MES, pues la mayoría de ésta no se utilizaría jamás.

En un desarrollo posterior de la AppMovil, puede ser necesaria una carga de proceso de datos importante, como podrían ser cálculo estadísticos sobre las producciones pasadas. Esta funcionalidad sería muy pesada para hacerla funcionar en los smartphones actuales, así, se le pediría el procesamiento de los datos al servidor, y éste devolvería a la AppMovil el resultado obtenido.

Por estos motivos, se necesita que la AppMovil sea un terminal más del servidor, incluso del MES. La naturaleza de diseño del MES no está pensada para proporcionar la información que necesita la AppMovil.

5.2. El servidor

La AppServer es el software desarrollado en el servidor. Antes de entrar en detalle en la AppServer, cabe destacar qué es el servidor.

El servidor es la máquina que está en **comunicación directa con la base de datos del MES**, y a la vez ofrece el servicio de acceso a esta información para el cliente (AppMovil).

En este proyecto, al trabajar con un MES “virtual”, se crea la necesidad de tener que montar el propio servidor en el que trabajará el MES. Esto implica la instalación de un servidor, un servidor web y el gestor de bases de datos.

El servidor será un ordenador personal de bajas prestaciones, por lo cual, el sistema operativo debe ser liviano. Se instalará un sistema Linux, con la distribución Lubuntu.

La base de datos sobre la que se accederá a la información se tiene que crear, con la misma estructura que lo haría un MES. Se escoge instalar un gestor de bases de datos relacional, MySQL. Es un software fácil de instalar y mantener en un sistema Linux, además, el servicio está totalmente integrado con el resto de componentes del servidor que se utilizarán.

El servidor web será el encargado de recibir las peticiones y entregar la información deseada a la AppMovil. La plataforma más extendida actualmente en el mercado es Apache, un **servidor web HTTP** de código abierto. Una de las ventajas de su uso es la facilidad de acceso a la documentación y recursos existentes.

Por último, el lenguaje de programación a utilizar en el servidor será **PHP**. Este lenguaje fue diseñado en sus orígenes para el desarrollo web de contenido dinámico, integrando el acceso a bases de datos de manera sencilla. PHP está integrado directamente en el servidor web Apache. En otros lenguajes de programación es necesaria la instalación de paquetes adicionales para su correcto funcionamiento, como podría ser el caso del lenguaje Python, con el que se ha programado la AppMovil.

El conjunto de infraestructuras Linux, Apache, MySQL y PHP recibe el nombre de servidor LAMP.

5.3. Protocolo de comunicación. HTTP.

Como se ha comentado, la AppMovil y la AppServer se comunicarán entre sí. Igual que pasa con las personas, para poder comunicarse, se deben cumplir una serie de normas y estructuras.

El protocolo de comunicación es la norma que rige **cómo se relacionará la AppMovil con la AppServer**.

La intercomunicación de datos a través de internet utiliza el conjunto de protocolos TCP/IP (Transmission Control Protocol / Internet Protocol). Dentro de este conjunto, encontramos una variedad de protocolos existentes, por ejemplo: FTP, POP, HTTP, SMTP o Telnet; cada uno de ellos diseñado para una funcionalidad concreta.

La cantidad de información que se transmitirá entre AppMovil y AppServer será de un tamaño pequeño en cada petición. El protocolo adecuado para este tipo de comunicación es HTTP, el utilizado por el servidor web.

El servidor web se rige por el protocolo HTTP (Hypertext Transfer Protocol). En él se define la sintaxis y semántica que utiliza el software para comunicarse. Es un protocolo pensado para transacciones del tipo petición-respuesta entre un cliente-servidor. [6]

El cliente que efectúa la petición recibe el nombre de *User Agent*. A la **información transmitida recibe el nombre de *URL*** *(Uniform Resource Locator)*. En el caso del proyecto, la AppMovil será el *User Agent*, y un ejemplo de *URL* sería (más adelante se entrará en detalle):

$$http://servidor\_web/json.php?peticion = bonos\&limit = 5$$

Este tipo de peticiones HTTP, se denominan **peticiones GET**.

Las diferentes peticiones que existen en el protocolo HTTP son: HEAD, GET, POST, PUT, DELETE, TRACE, OPTIONS y CONNECT. Cada una de estas peticiones indica la acción que se desea que efectúe el servidor web.

En la petición GET anterior, se pide al servidor que represente el recurso json.php con los parámetros petición y limit especificados.

Los servidores web entregan la información de cada petición en formato de **texto plano**, es decir, cadenas de caracteres encadenadas una tras otra. En el caso de las páginas web que se está acostumbrado a visitar, el navegador es el encargado de traducir el texto plano que entrega el servidor y mostrarlo de forma gráfica al usuario.

La ejecución del AppServer (json.php) representará la respuesta que el servidor web devuelva a la AppMovil. En la Fig. 17 se muestra la representación del recurso json.php.

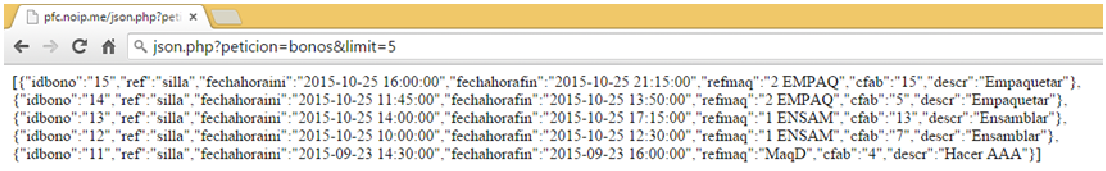


Fig. 17. Respuesta del servidor web a la petición del recurso json.php

5.4. Estructura de datos. JSON.

Como se ha comentado, el servidor web entrega texto plano, es decir, cadenas de caracteres encadenadas una tras otra. Este texto debe tener una estructura determinada para posibilitar el intercambio de datos entre cliente y servidor.

Existen **multitud de tipos de estructuras** para texto plano que organizan de manera ordenada e legible los datos. Aún así, se podría optar por un diseño estructural de datos específico entre la AppMovil i la AppServer. Algunos de los tipos estandarizados son: XML, JSON, YAML, Atom o EDN. En la Fig. 18 se observa como representar la misma información en los formatos XML, JSON y YAML, los más conocidos.

XML	`<catalogo>`
	`  <libro>`
	`  <autor>Raúl González Duque</autor>`
	`  <titulo>Python para todos</titulo>`
	`  <genero>Computación</genero>`
	`  </libro>`
	`</catalogo>`

JSON	`{`
	`  libro: {`
	`  autor: Raúl González Duque,`
	`  titulo: Python para todos,`
	`  genero: Computación`
	`  }`
	`}`

YAML	`---`
	`  libro:`
	`  autor: Raúl González Duque`
	`  titulo: Python para todos`
	`  genero: Computación`

Fig. 18. Formas de expresar la misma información en diferentes formatos.

Algunos de estos lenguajes son más simples que otros, además, pueden ser fácilmente legibles y entendidos a simple vista. Es el caso de JSON y YAML.

Existen **módulos prefabricados para integrar las estructuras de datos JSON** en los lenguajes de programación Python (utilizado en la AppMovil) y PHP (utilizado en la AppServer).

Los motivos anteriores hacen que JSON sea el formato de intercambio de datos elegido entre AppMovil y AppServer. [7]

La estructura del formato JSON está formada por un conjunto de **pares nombre/valor** separados por comas (,) y encerrados por llaves ({ }). Ver el cuadro JSON de la Fig. 18.

Los valores pueden ser únicos o una lista de valores. En caso que sean una lista de valores, se representan separados por comas (,) y encerrados por corchetes ([ ]). Ver Fig. 19.

JSON	{ libro: { autor: Raúl González Duque, titulo: Python para todos, genero: [Computación, Informática, Programación] } }

Fig. 19. Ejemplo de formato JSON utilizando listas de valores.

5.5. Programación

Como se ha comentado, el lenguaje de programación para la AppServer es PHP.

PHP es un lenguaje de código abierto, especialmente **diseñado para el desarrollo web dinámico**. Es decir, la visualización de la página web dependerá de la programación en PHP.

La programación básica de las páginas web se realiza en **lenguaje HTML**. Una visualización básica sería la siguiente:

```
<html>
        <head>
                <title>Título de la página web</title>
        </head>
        <body>
                <p>Primer párrafo de la web</p>
        </body>
</html>
```

Fig. 20. Ejemplo de programación HTML.

El lenguaje PHP puede ser incrustado en una página HTML y de esta manera transformarla en una web dinámica, como se observa a continuación:

```
<html>
	<head>
		<title>Título de la página web</title>
	</head>
	<body>
		<?php
			$conexion = mysqli_connect("localhost", "user", "pass", "MES");
			$sql = "SELECT ref FROM artículos";
			$result = mysqli_result($conexion, $sql);
			$i=0;
			while($row = mysqli_fetch_assoc($result)) {
				$rawdata[$i] = $row;
				echo $rawdata[$i];
				echo "<br>"
				$i++;
			}
			mysqli_close($conexion);
		?>
	</body>
</html>
```

Fig. 21. Ejemplo de programación PHP incrustado en HTML.

En el ejemplo anterior, el código PHP está entre las etiquetas **<?php** y **?>**. En primer lugar se conecta a la base de datos (MES), alojada en el mismo servidor (localhost), con nombre de usuario (user) y contraseña (pass).

Se define la consulta a realizar en la variable ($sql) y a continuación se ejecuta, almacenando el resultado en la variable ($result). En el bucle **while** se leen los datos almacenados en la variable $result y se van escriben como si fuera código HTML (echo $rawdata[$i]; echo "<br>").

Este código nos genera una página web en la que se muestra los diferentes artículos almacenados en la tabla artículos de la base de datos MES.

5.6. Funcionamiento

Cuando la AppMovil necesita información de la base de datos, realiza una petición al servidor web. El **servidor web ejecuta la AppServer**, y devuelve los datos.

Por ejemplo, la AppMovil necesita la información de los últimos 5 bonos de trabajo. A través del protocolo de comunicación HTTP hace la siguiente petición:

$$http://servidor\_web/json.php?peticion = bonos\&limit = 5$$

Fig. 22. Ejemplo de petición a la AppServer.

La petición anterior le pide al servidor web que ejecute el archivo *json.php*, con los parámetros petición y limit. El archivo PHP es la AppServer.

Los parámetros (petición y limit) recibidos por la AppServer determinan qué hará. En este caso, petición=bonos y limit=5, hace que la AppServer busque en la base de datos los 5 últimos registros de bonos de trabajo.

Esta información, la AppServer la devuelve en forma de estructura de datos en formato JSON. Ver Fig. 17, pág. 27.

En la Fig. 22 se observan diferentes apartados en la URL:

- Por una parte, el protocolo de comunicación (http://).
- seguido de la dirección IP o nombre del servidor (servidor_web/).
- a continuación se indica el archivo a ejecutar por el servidor web (json.php)
- seguido del carácter (?) se indican las parejas de parámetro-valor (petición=bonos&limit=5). Los diferentes parámetros (*petición* y *limit*) y valores (*bonos* y *5*), que serán los utilizados por la AppServer para realizar la acción específica que se desee. Las diferentes parejas de parámetro=valor se separan con el carácter (&).

La AppServer está programada para cumplir con las necesidades actuales de la AppMovil. Las funciones que soporta son las relacionadas con el módulo de planificación y el módulo de producción.

En relación al módulo de planificación, la información requerida es referente a las **estaciones de trabajo** y las **operaciones de las órdenes de fabricación**. Por otro lado, también necesita actualizar información de las operaciones a la hora de **modificar la prioridad** de éstas o **asignar una operación a una estación de trabajo** concreta. En la

Fig. 23 se representa de forma gráfica las diferentes funciones del módulo de planificación en la AppServer.

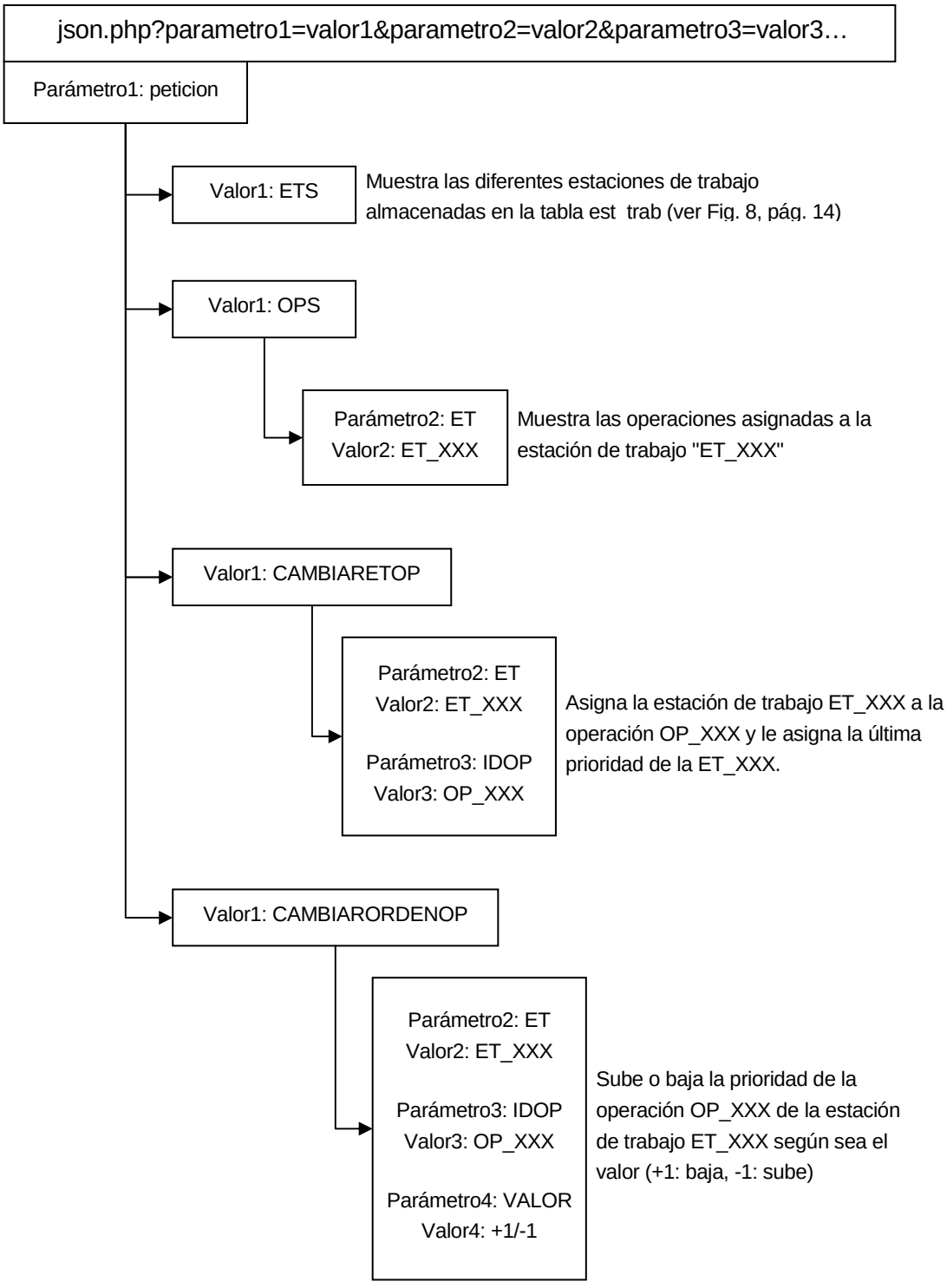


Fig. 23. Esquema de funcionamiento del módulo planificación de la AppServer.

El módulo de control y seguimiento de la producción tiene un uso diferente de la AppServer. En este caso se dispone de un solo tipo de *petición*, con una gran variedad de parámetros para determinar qué bonos de trabajo se quieres visualizar. En la Fig. 24 se observa el esquema de funcionamiento de los diferentes parámetros a tratar.

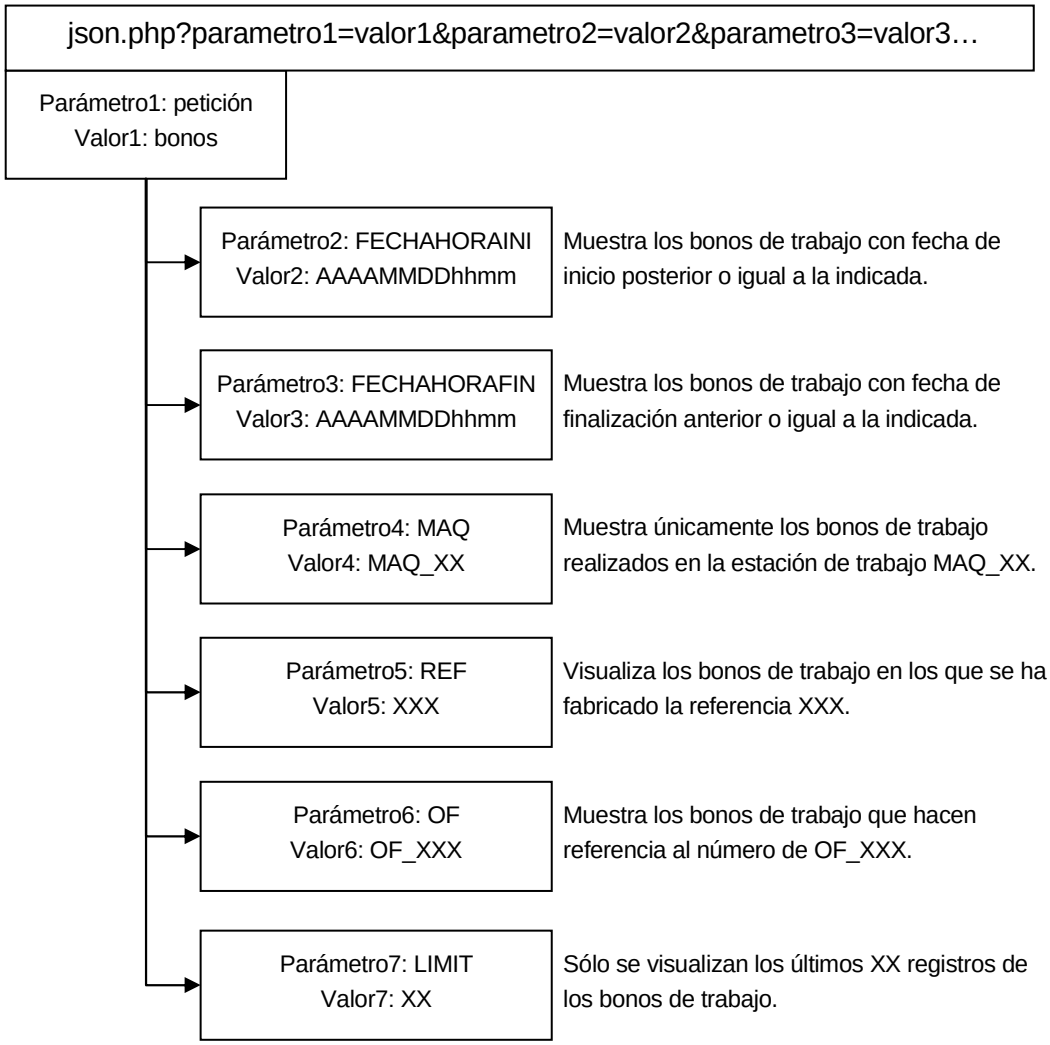


Fig. 24. Esquema de funcionamiento del módulo producción de la AppServer

6. Impacto Medioambiental

El medio ambiente es el conjunto de valores naturales, sociales, económicos y culturales existentes en un lugar y en un momento concreto. La puesta en marcha del proyecto implica una **interacción con los aspectos ambientales**, hecho que provoca unos efectos sobre ellos.

Como se ha comentado en el alcance del proyecto, éste es una primera versión de un software que tendrá otros impactos medioambientales. El impacto ambiental analizado será el causado por las funcionalidades del prototipo desarrollado durante este proyecto.

Los usuarios de la AppMovil tendrán la posibilidad de utilizarla cuando esté instalada en su smartphone y que tengan conexión a internet. Antes de la utilización de la AppMovil, estas personas podían realizar la planificación y seguimiento de la producción estando presente en la planta productiva, de esta manera, se obtiene una **ampliación del horario sobre el que se puede actuar sobre la producción**.

Un primer efecto económico que se desprende de este control intensivo radica en el **aumento de la capacidad de reacción ante imprevistos en la planificación**. Por ejemplo, un cliente anula un pedido que iba a ser fabricado en las próximas horas. De esta manera, se puede modificar la planificación de la máquina y anular la fabricación de ese producto que no podrá ser vendido de forma inmediata, provocando una **reducción de costes de almacenaje**, o incluso **reduciendo los desperdicios** en caso de no venderse un producto. La materia prima que no se ha consumido, incluso podría devolverse al proveedor.

Otra consecuencia de la capacidad de reacción vendría dada en caso que de **detectar una falta de materia prima** para realizar una producción, así, se conseguiría **evitar malgastar el tiempo** del operario en la preparación y ajuste de la máquina.

La facilidad de acceso a la AppMovil tiene un efecto social, puede provocar un **aumento de horas dedicadas al trabajo** y la **complicación de desconectar fuera del horario laboral**, por parte de los encargados de la gestión de la planta productiva.

La ampliación de la supervisión sobre la producción también afecta a las mermas productivas. Una **disminución de mermas** conlleva una reducción de costes y de residuos. Esta reducción de residuos tendrá efectos sobre la recogida, con la ampliación del periodo de tiempo entre dos recogidas de residuos consecutivas, con la correspondiente **reducción de emisión de gases** de su transporte.

La implantación de la AppMovil en la empresa genera una **nueva función en la empresa**, encargada del mantenimiento, incidencias y formación de la herramienta, que puede ser

decisiva para la contratación de nuevo personal. Esto será una ventaja para la persona contratada, y para la sociedad.

La posibilidad de acceder a los datos proporcionados por las producciones y el análisis de éstos, crea una **cultura de optimización de recursos**, que será beneficioso para la empresa a medio plazo, y afectará de forma indirecta a los trabajadores a largo plazo. La metodología de mejora de procesos por análisis de datos será un beneficio mutuo entre empresa y trabajadores.

Poder disponer de datos sobre la producción en tiempo real por parte del departamento comercial y de atención al cliente disminuiría el número de consultas que éstos realizan al departamento de operaciones. Reduciendo el número de llamadas de teléfono, consultas personales o correos electrónicos, liberando tiempo de trabajo a estos departamentos.

7. Planificación y Análisis económico

El proyecto se desarrolla en un marco temporal restringido por la cantidad de horas de trabajo que implica cada uno de los desarrollos que se han llevado a cabo.

A la vez, la ejecución del proyecto conlleva un coste debido a los diferentes factores que han entrado en juego, por una parte, recursos humanos, y por otra, recursos materiales.

En la Tabla 2 se detallan las tareas que se han llevado a cabo, analizando la carga temporal y el coste asociado a cada una, obteniendo de esta manera el coste total de los recursos humanos que han sido necesarios.

Tarea	Tiempo (h)	Precio/hora (€/h)	Coste (€)
Análisis funcionalidades y visualización general			
Consultor júnior	10	30	300
Consultor sénior	4	70	280
Configuración de servidor			
Consultor júnior	5	30	150
AppMovil			
Instalación y configuración			
Consultor júnior	5	30	150
Visualización			
Consultor júnior	10	30	300
Consultor sénior	1	70	70
Módulo planificación			
Consultor júnior	32	30	960
Consultor sénior	2	70	140
Módulo producción			
Consultor júnior	40	30	1.200
Consultor sénior	2	70	140
AppServer			
Consultor júnior	10	30	300
Consultor sénior	1	70	70
Total horas de consultor júnior	**112**	**30**	**3.360**
Total horas consultor sénior	**10**	**70**	**700**
Coste Total Recursos Humanos			**4.060**

Tabla 2. Coste de los recursos humanos del proyecto.

En la Tabla 3 se observan los diferentes costes materiales referentes al proyecto.

Concepto	Coste unitario (€)	Cantidad	Coste total (€)
Ordenador-Servidor	400,00	1	400,00
Smartphone	150,00	1	150,00
Varios (material, consumibles, etc.)	100,00	1	100,00
Coste Total Recursos Materiales			**750,00**

Tabla 3. Coste de los recursos materiales del proyecto.

Como se observa, el coste principal del proyecto recae sobre los recursos humanos. Esta es una característica del desarrollo de software.

Concepto	Coste total (€)
RECURSOS HUMANOS	4.060,00
RECURSOS MATERIALES	750,00
Coste Total Proyecto	**4.810,00**
IVA (21%)	1.010,10
Total Neto	**5.820,10**

Tabla 4. Costes del proyecto.

Cabe destacar de este análisis económico, que los costes analizados corresponden íntegramente al desarrollo de este proyecto, delimitado en el apartado 1.4. Alcance del proyecto.

El coste analizado no incluye la **implantación** del software en la empresa. Pues sería necesario un estudio previo del MES que hubiera en funcionamiento. Así como también se debería analizar la complejidad de la red de la empresa.

Junto con la implantación de un nuevo servicio en la empresa, es necesaria una **formación** de los futuros usuarios del servicio, en este caso la AppMovil. Este estudio tampoco está integrado en este análisis económico, pues depende de diferentes factores como el número de usuarios al que está destinado, así como del número de funcionalidades del software.

Aún así, se puede hacer una estimación del coste relativo a la formación de X usuarios para las funcionalidades desarrolladas en el proyecto. Es estima un tiempo fijo de 2h de trabajo para el desarrollo del manual de cada funcionalidad, en este caso, 2 funcionalidades. Un

tiempo de 30min por la formación presencial de cada grupo de 5 personas que utilicen el software. Y un coste de las instrucciones de uso de 0,50€ por cada usuario.

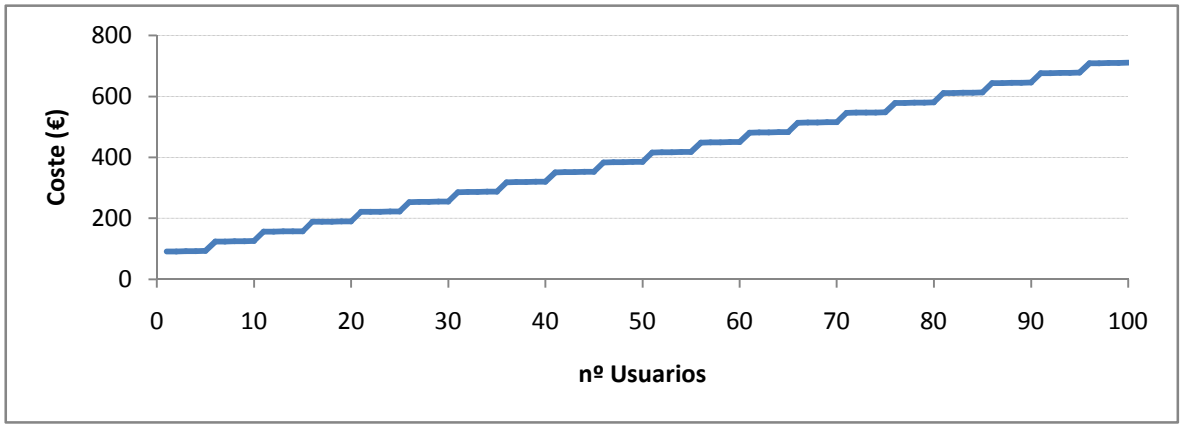


Fig. 25. Coste de formación en función del número de usuarios, para 2 funcionalidades.

Un último coste no incluido, es el coste de **mantenimiento** o servicio post-venta. En este tipo de coste incluyen parámetros como el número de dispositivos y la complejidad de la red de la empresa.

8. Siguientes pasos

El objetivo del proyecto ha sido desarrollar un prototipo de aplicación móvil, capaz de comunicarse con una aplicación servidor que proporcione los datos almacenados en una base de datos, como la que se utilizaría en una empresa con un sistema MES implantado.

Se ha comprobado la **viabilidad del prototipo**. Por lo tanto, se está en disposición de llevar a cabo el **desarrollo de software completo** que sea adecuado para una empresa en concreto.

Una vez la empresa esté dispuesta a implantar la AppMovil en su sistema MES, los pasos a seguir serían los siguientes.

En primer lugar, un **análisis del sistema** MES que la empresa tiene implantado. El objetivo será entender el funcionamiento de las bases de datos del MES y conocer la infraestructura que lo sostiene. Este hecho afectará a cómo recoge los datos la AppServer.

A continuación, se tendrán que especificar las **funcionalidades** a las cuales se desea tener acceso desde la AppMovil. Cada una de las funcionalidades se incluirá en alguno de los módulos definidos por la norma ISA-95 (Fig. 2, pág. 9): planificación, producción, gestión de materias primas, mantenimiento, calidad, etc.

El desarrollo de las diferentes funcionalidades supondrá una **carga de trabajo independiente** en la mayoría de los casos. En consecuencia, la cantidad de funcionalidades y la complejidad de desarrollarlas impactará directamente en el coste del software completo. Para tener una idea previa de este impacto, se estimará el tiempo de trabajo necesario para alguna funcionalidad que puede ser adecuada según la empresa.

Control del inventario de materias primas. Es común que una empresa utilice una materia prima (MP) para fabricar diferentes productos. En el caso que el stock de una MP no fuera suficiente para fabricar todas las OF que la necesitan, se podría asignar una cierta cantidad a cada OF según criterios no automáticos. El desarrollo es esta funcionalidad sería del orden de unas 30 horas de trabajo.

Gestión del mantenimiento. El mantenimiento de una máquina es fundamental para que sea fiable, reduciendo el número de averías y fallos que puede producir. Desarrollar una funcionalidad para gestionar el plan de mantenimiento a través de la AppMovil supone una carga de trabajo de unas 15 horas.

Control de calidad. Las piezas producidas por una máquina deben ser verificadas. En función del proceso productivo y el producto, existen diferentes criterios para determinar la

frecuencia de verificación y qué verificar en cada inspección. Una funcionalidad que permita hacer un seguimiento sobre qué controles de calidad se han realizado y cuál ha sido su resultado se estiman 50 horas de trabajo. En caso que la funcionalidad sea la gestión del sistema de control de calidad (modificar frecuencias de verificación), el tiempo sería de 20 horas.

Se observa que la carga de trabajo en función del desarrollo a realizar es dispar. Por lo tanto, es necesario llegar a un equilibrio coste/beneficio de cada funcionalidad para determinar si es necesario implantarlo en la empresa.

9. Planificación de la implantación

En el apartado anterior se ha comentado los siguientes pasos a realizar para implantar la solución de software completo en una empresa.

A continuación, se situará en un espacio temporal las diferentes acciones a realizar, incluyendo en el calendario otras restricciones que aparecen en el momento que analizar los factores coste-beneficio.

A la hora de implantar el software en la empresa, es necesario negociar la vinculación que se adquiere con esta y el coste que supondrá la implantación y posterior mantenimiento.

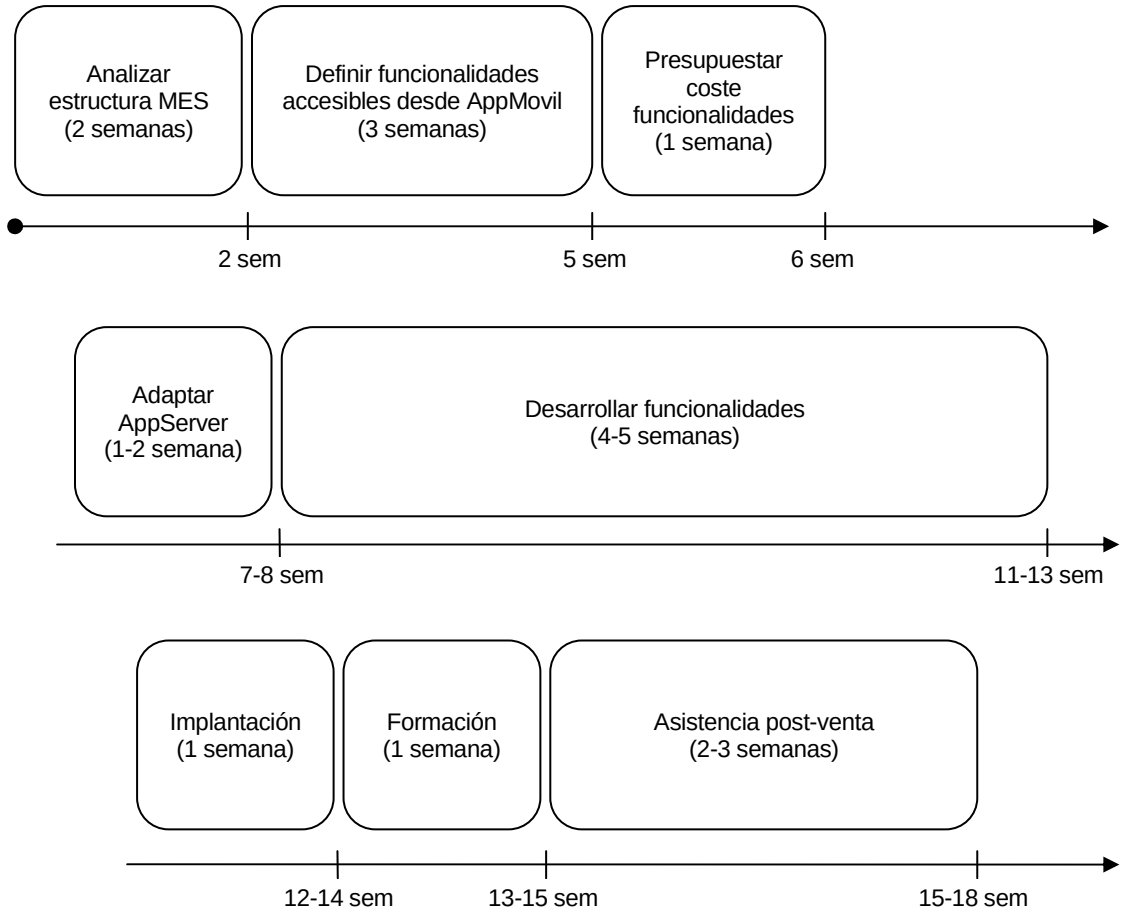


Fig. 26. Planificación de la implantación del software completo.

En la figura anterior se observa la planificación temporal de la implantación de un software completo, teniendo en cuenta el desarrollo de las funcionalidades que se comentaron en el

apartado anterior: control del inventario de materias primas, gestión del mantenimiento y control de calidad.

El tiempo transcurrido desde la decisión de querer utilizar la AppMovil hasta la implantación sería entre 12 y 14 semanas, unos 3 meses.

Conclusiones

En el mundo empresarial se está implantando el **trabajo a distancia**, sin la necesidad restricciones temporales ni espaciales. La **conciliación** de la vida familiar con la laboral es una realidad que se debe tener en cuenta a la hora de diseñar las empresas del futuro.

Para hacer posible esta **desvinculación** del lugar del trabajo se han utilizado multitud de recursos tecnológicos que lo hacen posible. Los ordenadores portátiles, internet, la nube, los smartphones, las tablets o los weareables tan sólo son un ejemplo de todos los avances que nos ofrecen **movilidad**.

Durante el desarrollo de proyecto se ha visto que el prototipo de **AppMovil conectada con la AppServer para la gestión de un software MES es viable**. Este software contribuye a facilitar esta desvinculación con el lugar de trabajo, mejorando la vida familiar de los usuarios de los sistemas MES.

Las **funcionalidades** que se determinen incorporar a la AppMovil deben ser previamente **analizadas por los diferentes actores** que las utilizarán. El coste de desarrollo de cada funcionalidad es asequible para las empresas, aún así, se debe tener en cuenta que cada funcionalidad debe tener un objetivo claro. La idea del vamos a incorporar esta funcionalidad "por si acaso", rara vez acaba teniendo éxito. Es preferible dedicar un mayor esfuerzo a definir cuáles son los **objetivos a medio-largo plazo de la planta productiva**, y adaptar los sistemas de información a tales objetivos.

Es importante tener en cuenta que la tecnología es un recurso del que se dispone para mejorar la calidad de vida de las personas. Deben ser las personas las que dominen la tecnología, y no dejarse llevar por ella.

La **usabilidad** de una aplicación móvil es una prioridad a la hora de desarrollarla. Una gran parte de de la usabilidad depende de la **experiencia del usuario**. Si la App causa buena sensación en la primera impresión la aceptación aumenta. Para esto es importante contar con un buen **asesoramiento gráfico** a la hora de desarrollarla.

La programación de la AppMovil se ha realizado con el lenguaje de programación **Python**. En concreto, con el **framework Kivy**, una especialización de Python para la programación de aplicaciones móviles. El uso de este lenguaje ha sido una novedad y una grata sorpresa. Kivy incorpora un propio lenguaje de programación llamado Kv-language, destinado al diseño gráfico de las aplicaciones. El **aprendizaje** tanto de este lenguaje como de python, totales desconocidos antes del desarrollo del proyecto, ha sido posible gracias a la cantidad de **información disponible**.

Agradecimientos

Me gustaría agradecer la realización de este proyecto a todas esas personas que me han permitido adquirir conocimientos en el área de las tecnologías de la información, así a aquellos que han motivado y apoyado a lo largo de la trayectoria profesional.

Agradezco el seguimiento y ilusión mostrada por el tutor del proyecto, Lluís Solano. Que también durante el periodo que me impartió docencia amplió mis conocimientos informáticos.

Por último, agradecer a mi familia, pareja y amigos el apoyo mostrado durante tantos años. Gracias a todos.

Bibliografía

Referencias bibliográficas

[1] CINCOMASAPP. La usabilidad [http://cincomasapp.com/usabilidad-y-apps-usables, mayo de 2015]

[2] KANTAR WORLD PANEL. Ventas de smartphones. [http://www.kantarworldpanel.com/es/Noticias/Ventas-de-smartphones-julio-Android-roza-el-90-del-mercado, septiembre de 2015]

[3] ANDROID. Ayuda al desarrollador. [http://developer.android.com/sdk/installing/index.html?pkg=tolos, abril de 2015]

[4] DESARROLLOWEB. MVC. [http://www.desarrolloweb.com/articulos/que-es-mvc.html, septiembre de 2015]

[5] DRUPALALSUR. Instalación de servidor LAMP. [http://drupalalsur.org/apuntes/como-instalar-un-servidor-lamp-en-ubuntu-1404, abril de 2015]

[6] UNIVERSITAT DE VALENCIA. Departament d'Informàtica [http://informatica.uv.es/iiguia/IST/Tema2.pdf, agosto de 2015]

[7] JSON. Introducción a JSON. [http://www.json.org/json-es.html, agosto de 2015]

Bibliografía complementaria

CODECADEMY. Tutorial de Python. [https://www.codecademy.com/es, abril de 2015]

STACKOVERFLOW. Kivy questions. [http://stackoverflow.com]

DESARROLLOWEB. Tutorial PHP. [http://www.desarrolloweb.com/php/, julio de 2015]

TONYCODE. HTTP requests. [http://blog.tonycode.com/tech-stuff/http-notes/making-http-requests-via-telnet, septiembre de 2015]

GITHUB. Various kivy sources. [https://github.com/]